CIS 24 - Programming Languages - Midterm
Name:

1. In what ways are high-level languages an improvement on assembly language? In what circumstances does it still make sense to program in assembler?

2. On modern machines, do assembly language programmers still tend to write better code than a good compiler produces? Why or why not?

3. What distinguishes declarative languages from imperative languages?

4. Explain the distinction between interpretation and compilation. What are the comparative advantages and disadvantages of the two approaches.

5. List the principal phases of compilation, describe the work performed by each. What abstract machine is implemented by the scanner? Parser?

6. Describe the form in which a program is passed from scanner to the parser; from the parser to the semantic analyzer; from the semantic analyzer to the intermediate code generator.

7. What is the purpose of a compiler's symbol table?

8. What is the difference between syntax and semantics?

9. What are the three basic operations that can be used to build complex regular expressions from simpler regular expressions?

10. What additional operation (beyond the three of regular expressions) is provided in context-free grammars?

11. What is the difference between a right-most derivation and a left-most derivation? Give an example.

12. What does it mean for a context-free grammar to be ambiguous? Give an example.

13. What are associativity and precedence?

14. Summarize the difference between LL and LR parsing. Which one of them is called "bottom-up"? "Top-down"? Which one of them is also called "predictive"? "Shift-reduce"? What do "LL" and "LR" stand for?

15. Describe two common idioms in context-free grammars that cannot be parsed top-down. Why not? Give an example.

16. What is *binding time*? What do we mean by the *scope* of a name-to-object binding?

17. What determines whether a language rule is a matter of syntax or of static semantics?

18. Some compilers perform all semantic checks and intermediate code generation in action routines. Others use action routines to build a syntax tree and then perform semantic checks and intermediate code generation in separate traversals of the syntax tree. Discuss tradeoffs between these two strategies.

19. Define an LL(1) grammar for arithmetic expressions. (Note that the grammar must not have left recursion, nor common prefixes)

20. Consider the following grammar (Figure 1):

$$G \rightarrow S\,\$\$$$

$$S \rightarrow A\,M$$

$$M \rightarrow S \mid \varepsilon$$

$$A \rightarrow a\,E \mid b\,A\,A$$

$$E \rightarrow a\,B \mid b\,A \mid \varepsilon$$

$$B \rightarrow b\,E \mid a\,B\,B$$

Figure 1: Context Free Grammar

(a) Describe in English the language that the grammar generates.
(b) Show a parse tree for the string: $a\,b\,a\,a$
(c) Is the grammar LL(1)? Why or why not?